The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. In a computer system having at least one software program operable to instantiate objects having at least one property, a method for processing object property changes, the method comprising:

obtaining a property change for a source object property change, wherein a source property change value is not dependent on any other object properties;

obtaining a pre-source change steady-state value for at least one object property dependent on the source object property, wherein a dependent object property value is determined by an evaluation of a relational expression;

obtaining a post-source change steady-state value for at least one object property dependent on the source object property;

processing any non-affected dependent object properties;

processing any duplicative dependent object properties; and

implementing any remaining dependent object properties.

2. The method as recited in Claim 1, wherein obtaining the source object property change includes obtaining from the application program a value change for the source object property.

3. The method as recited in Claim 1, wherein obtaining the pre-source change steady-state value includes evaluating each dependent object property relational expression to obtain an object property value.

4. The method as recited in Claim 1, wherein obtaining the pre-source change steady-state value includes obtaining at least one cached object property value, wherein cached object property value corresponds to a previously evaluated relational expression.

5. The method as recited in Claim 1, wherein obtaining the post-source change steady-state value includes evaluating each dependent object property relational expression to obtain an object property value.

6. The method as recited in Claim 1 further comprising, prior to processing any non-affected dependent object properties:

generating a memory array containing array elements corresponding to the source object property and at least one dependent object property; and

populating the array elements of the memory array with the pre-source change steady-state values for all object properties represented in the memory array.

7. The method as recited in Claim 6, wherein generating the memory array includes, for each array element in the array, generating an array element for each object property that depends from an array element, wherein each object property is linked to its dependent properties and wherein an object property may be represented by more than one array element.

8. The method as recited in Claim 6 further comprising correlating back-dependency data for each array element in the memory array, wherein the back dependency data corresponds to array elements related to the same object.

9. The method as recited in Claim 6 further comprising populating the array elements of the memory array with the post-source change steady-state values for all object properties represented in the memory array.

10. The method as recited in Claim 9, wherein processing any non-affected dependent object properties includes, for each array element, determining whether the object property pre-source change steady-state value is equal to the object property post-source change steady-state value.

11. The method as recited in Claim 10, wherein processing any non-affected dependent object properties includes voiding any array element whose object property pre-source change steady-state value is equal to the object property post-source change steady-state value.

12. The method as recited in Claim 11, wherein processing any non-affected dependent object properties includes voiding any array elements linked to a voided array element.

13. The method as recited in Claim 12, wherein processing any duplicative dependent object properties includes, for every non-void array element:

determining whether any array elements in the memory array correspond to the same object property; and

if one or more array elements correspond to the same object property, determining the most recent array element corresponding to the object property; and

voiding any additional array elements corresponding to the object property.

14. The method as recited in Claim 13, wherein determining the most recent array element includes iteratively comparing two array elements and voiding the least recent of the two array elements.

15. The method as recited in Claim 1, wherein implementing any remaining dependent object properties includes instantiating one or more objects with the post-source change object property values.

16. The method as recited in Claim 1, wherein the objects include user interface objects.

17. A computer-readable medium having computer-executable instructions operable for performing the method as recited in Claim 1.

18. A computer system having a processor and a memory, the computer system operable to perform the method as recited in Claim 1.

19. In a computer system having a software program operable to generate one or more user interfaces having a number of user interface objects and at least one software application operable to instantiate user interface objects having at least one property, a method for processing user interface object property changes, the method comprising:

obtaining a property change for a source user interface object property change, wherein a source property change value is not dependent on any other user interface object properties;

obtaining a pre-source change steady-state value for at least one user interface object property dependent on the source object property, wherein a dependent user interface object property value is determined by an evaluation of a relational expression;

generating a memory array containing array elements corresponding to the source user interface object property and at least one dependent user interface object property; and

populating the array elements of the memory array with the pre-source change steady-state values for all user interface object properties represented in the memory array.

obtaining a post-source change steady-state value for at least one user interface object property dependent on the source user interface object property;

populating the array elements of the memory array with the post-source change steady-state values for all user interface object properties represented in the memory array;

coalescing the array elements of memory array; and

implementing any remaining dependent user interface object properties.

20. The method as recited in Claim 19, wherein obtaining the source object property change includes obtaining from the application program a value change for the source object property.

21. The method as recited in Claim 19, wherein obtaining the pre-source change steady-state value includes evaluating each dependent object property relational expression to obtain an object property value.

22. The method as recited in Claim 19, wherein obtaining the pre-source change steady-state value includes obtaining at least one cached object property value, wherein cached object property value corresponds to a previously evaluated relational expression.

23. The method as recited in Claim 19, wherein obtaining the post-source change steady-state value includes evaluating each dependent object property relational expression with a to obtain an object property value.

24. The method as recited in Claim 19, wherein generating the memory array includes, for each array element in the array, generating an array element for each object property that depends from an array element, wherein each object property is linked to its dependent properties and wherein an object property may be represented by more than one array element.

25. The method as recited in Claim 19 further comprising correlating back-dependency data for each array element in the memory array, wherein the back dependency data corresponds to array elements related to the same object.

26. The method as recited in Claim 19, wherein coalescing the array elements of the memory array includes, for each array element, determining whether the object property pre-source change steady-state value is equal to the object property post-source change steady-state value.

27. The method as recited in Claim 26, wherein coalescing the array elements of the memory array includes voiding any array element whose object property pre-source change steady-state value is equal to the object property post-source change steady-state value.

28. The method as recited in Claim 27, wherein processing any non-affected dependent object properties includes voiding any array elements linked to a voided array element.

29. The method as recited in Claim 19, wherein coalescing the array elements of the memory array includes, for every non-void array element:

determining whether any array elements in the memory array correspond to the same user interface object property; and

if one or more array elements correspond to the same user interface object property, determining the most recent array element corresponding to the user interface object property; and

voiding any additional array elements corresponding to the user interface object property.

30. The method as recited in Claim 29, wherein determining the most recent array element includes iteratively comparing two array elements and voiding the least recent of the two array elements.

31. The method as recited in Claim 19, wherein implementing any remaining dependent user interface object properties includes instantiating one or more user interface objects with the post-source change user interface object property values.

32. A computer-readable medium having computer-executable instructions operable for performing the method as recited in Claim 19.

33. A computer system having a processor and a memory, the computer system operable to perform the method as recited in Claim 19.

34. In a computer system having at least one software application operable to instantiate objects having at least one property, a method for processing object property changes, the method comprising:

obtaining a property change for a source object property change, wherein a source property change value is not dependent on any other object properties;

obtaining a pre-source change steady-state value for at least one object property dependent on the source object property, wherein a dependent user interface object property value is determined by an evaluation of a relational expression;

generating a memory array containing array elements corresponding to the source object property and at least one dependent object property;

for each array element in the array, generating additional array elements for each object property that depends from an array element, wherein each object property is linked to its dependent properties and wherein an object property may be represented by more than one array element;

populating the array elements of the memory array with the pre-source change steady-state values for all object properties represented in the memory array;

obtaining a post-source change steady-state value for at least one object property dependent on the source object property;

populating the array elements of the memory array with the post-source change steady-state values for all object properties represented in the memory array;

performing a breadth-first array iteration to eliminate any non-affected array elements;

performing a depth-first array iteration to eliminate any duplicative array elements; and

implementing any remaining dependent user interface object properties.

35. The method as recited in Claim 34, wherein obtaining the source object property change includes obtaining from the software application program a value change for the source object property.

36. The method as recited in Claim 34, wherein obtaining the pre-source change steady-state value includes evaluating each dependent object property relational expression to obtain an object property value.

37. The method as recited in Claim 34, wherein obtaining the pre-source change steady-state value includes obtaining at least one cached object property value, wherein cached object property value corresponds to a previously evaluated relational expression.

38. The method as recited in Claim 34, wherein obtaining the post-source change steady-state value includes evaluating each dependent object property relational expression with a to obtain an object property value.

39. The method as recited in Claim 34 further comprising correlating back-dependency data for each array element in the memory array, wherein the back dependency data corresponds to array elements related to the same object.

40. The method as recited in Claim 34, wherein performing a breadth-first array iteration includes, for each array element, determining whether the object property pre-source change steady-state value is equal to the object property post-source change steady-state value.

41. The method as recited in Claim 40, wherein performing a breadth-first array iteration includes voiding any array element whose object property pre-source change steady-state value is equal to the object property post-source change steady-state value.

42. The method as recited in Claim 41, wherein performing a breadth-first array iteration includes voiding any array elements linked to a voided array element.

43. The method as recited in Claim 34, wherein performing a depth-first array iteration includes, for every non-void array element:

determining whether any array elements in the memory array correspond to the same object property; and

if one or more array elements correspond to the same object property, determining the most recent array element corresponding to the object property; and

voiding any additional array elements corresponding to the object property.

44. The method as recited in Claim 43, wherein determining the most recent array element includes iteratively comparing two array elements and voiding the least recent of the two array elements.

45. The method as recited in Claim 34, wherein implementing any remaining dependent object properties includes instantiating one or more objects with the post-source change object property values.

46. The method as recited in Claim 34, wherein the objects include user interface objects.

47. A computer-readable medium having computer-executable instructions operable for performing the method as recited in Claim 34.

48. A computer system having a processor and a memory, the computer system operable to perform the method as recited in Claim 34.